Data Compression and VLSI Implementation

Wai-Chi Fang

Jet Propulsion Laboratory, California Institute of Technology

Radar Science and Engineering Section

MS 300-241,4800 Oak Grove hr., Pasadena CA 91109-8099

(818) 354-4695, c-mail: wfang@blacks.jpl.nasa.gov

Abstract --- VLSI data compression research has been motivated by the needs of high-speed high-performance data compression and inspired by the VLSI technologies, An integrated data compression system has been proposed to provides adaptive multi-mode data compression for an advanced multi-instrument spacecraft payload system that has various source data. It combines a high-ratio lossy data compressor with a lossless data compressor to provide various compression schemes. The embedded lossy compressor is a high-speed neuroprocessor for adaptive vector quantization based upon a frequency-sensitive self-organization neural algorithm, The embedded lossless compressor is a high-speed pipelined processor design based on the Rice algorithm.

1 Introduction

1)ata compression is essential in reducing the data transmission or storage costs for broad areas of applications such as high-definition television, teleconferencing, remote sensing, radar, sonar, computer communication, facsimile transmission, image database management, and science instrument. in the research field of data compression, important contributions have come from information sciences, computer sciences, signal and image processing, computer engineering, very IarCc-scale integration (VLSI) technologies, etc. [1-8]. A new trend of this research appears to have shifted from basic research to system development, in particular to high-speed VLSI implementation of image and video coders [6-8]. The driving forces behind this trend arc as follows. (a) Advances of electronic technologies, especially dramatic progress in VLSI technologies, make more sophisticated data compression systems possible. (b) The increasingly strong demands on large-volume data communication and storage requirements in Informa t ion Age. (c) Several key driving applications require high-performance data compression techniques such as high-definition TV, advanced multi-media information systems, integrated service data networks, satellite imaging system, and the space science data and communication systems.

VLSI data compression research involves a very broad spectrum of disciplines, including system analysis, simulation of the processing algorithms, extraction of parallel /pipeline computing structure, development of processor archi t ecture, design of functional and structural primitives, layout generation of VLSI chips, fabrication, packaging, testing of functional and electrical properties, reliability and qualification. More efforts arc still need cd to develop high-speed, high-performance VI .SI data compression systems. Improvements in the data compression algorithm, architecture, software and hardware implementation, and associated system-level design issues are critical to effectively achieve the goal of data compression,

This tutorial paper presents the algorithms, architectures and VLSI processors used in an integrated data compression system. Section two describes the proposed integrated data compression system. Section three describes the embedded lossless data compression processor. Section four describes the embedded lossy image compression processor.

2 integrated 1)ata Compression System

An integrated data compression system has been proposed to meet the needs of high-speed high-performance data compression. 1 ts primary application is targeted for the advanced multi-instrument spacecraft payload system that has various source data such as science data, image, video, and binary files [21]. Among these various data types, the video and image data is the driver

of the }~i~h-speed high-performance data compression due to the real-time high-fidelity requirements and the large data volume involved. Figure 1 shows a functional diagram of the adaptive multi-mock data compression system that combines a high-ratio lossy data compressor with a , 1o ssIcss data compressor to provide various compression schemes to meet a broad area of applications for various instruments.

The features of the proposed adaptive multi-instrument data compression system are summarized as follows:

(A) The proposed multi-mode data compression scheme is reconfigurable to support five different operation modes for various instruments: (a) bypass (no compression), (b) lossy compression, (c) 1os1css compression, (d) lossy compression enhanced with lossless compression, and (e) lossless compression enhanced with lossy compression. The lossy compression portion is capable of producing good reconstructed data quality at high compression ratios while the lossless compression portion can produce reconstructed data without any fidelity loss. The reconfiguration signal *S1*, S2, and S3 arc from the communication processor. The selection of these compression modes is dependent on the inherent nature of the targeted applications. The effective maximum link or storage capability can be approximately increased by the same ratio as the achieved compression ratio over that of the baseline system.

(B) 1! has a communication co-processor to perform the channel coding and variable-length packetization and thus provide a stringent error-protected data quality and a flexible buffering management of the variable length data.

(C) It is a special-purpose high-speed signal compressor design with a state-of-art VLSI technologies and multichip module packaging to meet the computation intensive and real time processing requirements at a low mass, volume, and power consuming.

(D) The applications of the proposed multi-mode compression system cover a wide spectrum such as lossless medical image compression, lossless scientific data compression, audio and video data compression of high definit ion TV, multi-medium data compression, and other data communication anti/or storage applications.

The embedded lossy compressor and 1o ssIcss compressor are described in the following sections,

3 Neural-Network Based Lossy Data Compressor

A modified self-organization neural network algorithm and its associated VLSI neural processor have been developed for adaptive vector quantization [9]. Section 3.1 describes the frequency-sensitive self-orgallizatioll algorithm and system-level analysis results. Section 3.2 presents a massively paralleled VLSI neural network hardware to implement this algorithm. Section 3.3 discusses the detailed circuit design of the neural network processor chip. Section 3.4 presents the prototype chip.

3.1 The FSO Neural Algorithm

The frequency-scmsitive self-organization (FSO) method modifies competitive learning method [11,1 2] by applying a winning frequency and its associated upper-threshold value to the cmtroid-based learning rule. The FSO method is a fast and powerful scheme for adaptive vector quantization duc to its relatively low computing requirement and massivel y paralleled computing structure. It systematically distributes the codevectors in the vector space R^n to approximate the unknown probability density function of the training vectors. Codevectors quantize the vector space and converge to cluster centroids. Use of the upper-threshold frequency (F_{th}) can avoid codevector under-utilization during the training process for an inadequately chosen initial codebook. Empirically, an adequate F_{th} is chosen to be two to three times larger than the average winning frequency. The performance of the one-iteration FSO method can be incrementall y improved by using iteration to adjust codevectors into better cluster centroids. The codebook obtained from the previous iteration is used as the initial values of the current iteration. After the first iteration, the upper-threshold frequency is not needed, because a good initial codebook is available. This method is called the multiple-iteration FSO method. As shown in Fig. 2, performance of the FSO method is very close to that of the LBG method [10].

The FSO method for adaptive vector quantization is described as follows:

1) Initialize the codevectors $\mathbf{W}_i$ and the winning frequency F_i for each distortion-computing neuron:

$$\mathbf{W}_i(0) = \mathrm{R}(i), \qquad F_i(0) = 1, \; i = 1, \ldots, N, \tag{1}$$

where R(.) is a random vector-number generation function, M is the number of vector components, N is

the number of codevectors, and $\mathbf{W}_i(0) = [W_{i1}(0), W_{i2}(0), \ldots, W_{iM}(0)]$. Notice that the first N input vectors can also be used as the initial codevectors instead of using results generated from R(.).

2) Compute the distortion $D_i(t)$ between an input vector $\mathbf{X}(t)$ and all codevectors:

$$D_i(t) = d(\mathbf{X}(t), \mathbf{W}_i(t)) = \sum_{j=1}^{M} (X_j(t) - W_{ij}(t))^2, \quad (2)$$

where t is the training time index.

3) Select the distortion-computing neuron with the smallest distortion and set its Output $O_i(t)$ to high:

$$O_i(t) = \begin{cases} 1 & \text{if } D_i(t) < D_j(t),\ 1 \le i, j \le N,\ i \ne j, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

4) Update the codevectors with a frequency-sensitive training rule and the associated winning frequency:

$$\mathbf{W}_i((+1) = \mathrm{W},(\mathrm{l}) + \mathrm{S(l)}\ O_i(t)[\mathbf{X}(t) \mathbf{W}_i(t)], \quad (4)$$

$$S(t) = \begin{cases} \frac{1}{F_i(t)} & \text{if } 1 \le F_i(t) \le F_{th}, \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

$$F_i(t+1) = \mathrm{F,(f)} + O_i(t), \quad (6)$$

where $S(t)$ is the frequency-sensitive learning rate, and F_{th} is the upper-thrcshokl frequency. Only the winning codevector is updated. The training rule moves the winning codevector toward the training vector by a fractional amount which decreases as the winning frequency increases. If F_i is larger than F_{th}, then set S(f) to zero and no further training will be performed for this neural unit.

5) Repeat steps 2 through 4 for all training vectors.

3.2 VLSI Neural Processor Architecture

The proposed VLSI neurocomputing architecture for adaptive image compression using the frequency-sensitive self-organization network is shown in Fig. 3. The FSO network consists of two layers: an input layer and a competitive layer. The input layer consists of M input neurons, which correspond to the elements of the M-dimensional input vector. Each input neuron gets its input from the external data bus and distributes the buffered signal to N distortion-computing neural uni ts in the competitive layer through the synapse matrix. Each distortion-computing neuron calculates a squaw of Euclidean distance between its codevector and the input vector. The competitive process is performed throughout the whole layer by the winner-take-all operation. The winning neural unit is determined according to the minimum distortion criterion. The synapse weights are then updated according to the FSO learning rule as specified in (4), (5), and (6). A mixed-si~nal VLSI design technique is used to implement the paralleled vector quantizer. The analog circuitry performs massively paralleled neural computation and digital circuitry processes multiple-bit address information. This neural-based vector quantizer realizes a full-search vector quantization process for each input vector at a time complexity 0(1). '1 'he image data is accessed from the host computer through the interface and timing control block. The image data is divided into subimage blocks and can be stored in the dual-port vector memory. A subimage block is handled as a digital input vector. The digital input vector is converted into the analog value using the D-to-A converter array and fed to input neurons of the paralleled vector quantizer.

3.3 FSO Network Circuit

Figure 4 shows the functional block diagram of the FSO network slice consisting of key circuit blocks. The input neuron is a convent ional operational amplifier in a unity-gain configuration. The programmable synapse design is a modified wide-range Gilbert multiplier [22], which can perform real-valued multiplication in four quadrants and achieve 8-bit precision. The output summing neuron is a amplifier with a linear floating resistor that linearly converts the summed current up to 1mA into an analog voltage with the sign reversed, which is sent to the winner-take-all (WTA) circuit. One WTA cell consists of two portions. The first portion converts input voltage into the current which is compared and redistributed in the common signal line. The resultant input current in the winning ccl] is completely differentiated from all losing cells. in the second portion, the current is converted into the output voltage that provides fully binary output values to interfaced with digital circuitry for network learning. The simulated processing time for one network iteration is less than 500 nsec. Each iteration cycle includes input buffering, synapse multiplication, neuron summing, winner-take-a]l operation, and index encoding.

3.4 Prototype Chip

The prototype neural network processor chip for a 25-dimensional adaptive vector quantizer of 64 codevectors was fabricated in a silicon area of 46

mm x 6.8 mm using the 2-μm CMOS technology. This prototype chip includes 25 input neurons, 25 x 64 synapse cells, 64 distortion-computing neu inns, a winner-take-all circuit block, and a digital index encoder. 1 (s throughput rate is 2 million vectors per second and its equivalent computation power is 3.2 billion connections per second. 1 t achieved an intrinsic compression ratio of 33. The die photo of this ncwral-based vector quantizer chip is shown in Fig. 5. This chip can also be intended to implement vector quantization of a larger codebook. A n adaptive vector quantizer of 1024 codevectors can be implemented by cascading 16 such prototype chips or by using a larger design in a submicron fabrication technology.

4 Embedded Lossless Data Compressor

Lossless coding is a data compression technique that is employed to compress the data without inducing any distortion in the reconstructed data [3]. Lossless coding is also termed as noiseless coding, reversible coding, redundancy reduction coding, and entropy coding. Lossless coding is usually required in such situat ions where the compression system must be designed without prior knowledge of the structure or end use of the original data, Lossless coding is also suited as a post-processing stage for further reduction of data compressed by a lossy compression algorithm to maximize the compression to distortion noise ratio.

4.1 Universal Noiseless Coder

Among existing lossless data compression algorithms [15-20], Rice algorithm [15] is an effective method for producing reconstructed data without any fidelity loss. It provides excellent performance over a broad range of data contents. It is also an efficiently implementable scheme for VI .SI realization. The universal noiseless coding scheme in the Rice Algorithm is a subset of Huffman codes optimal for Laplacian symbol sets [20]. A gene'ric universal noiseless coder (UNC) is given in Fig. 6. It consists of three blocks: source data conditioner, block adaptive entropy coder (BAEC), and compressed data formatter. Figure 7 shows a functional design of PSI14 that is a specific version of UNC A high-speed UNC compressor chip has been implemented at a low hardware cost by using a VI .SI pipelined architecture [13,14].

4.2 Source Data Conditioner

The source data conditioner reversibly reformats the raw data source into the standard data source whose symbols can be represented by uncorrelated non-negative integers, where smaller integers arc more likely than that the larger ones.

The UNC design presumes that source data conditioning can be done by either an off-chip preprocessor or a cm-chip preprocessor. The function of an off-chip preprocessor can be arbitrary. The on-chip preprocessor is defined as a simple one-dimensional predictor and a mapper as specified in (7) and (8), respectively.

't'he prediction operation involves the operation of taking differences between consecutive pixels:

$$A = x(i) - x(i-1) \tag{7}$$

where $x(i)$ is the i th pixel and $x(i-1)$ is the i-1 th pixel. The 'prediction operation reduces the data entropy if- there is a high correlation among adjacent pixels.

The mapping step transforms an (n-l 1)-bit difference into an n-bit positive integer representation:

$$\delta = \begin{cases} 2\Lambda - - - 1 & , \quad 0 < \Lambda \le \theta \\ -2\Lambda & , \quad -\theta \le \Lambda \le 0 \\ \theta + |\Lambda| & , \quad |\Lambda| > \theta \end{cases} \tag{8}$$

where $\theta = MIN[x(i-1), 2^n - 1 - x(i-1)]$ and n is the number of bits per input pixel.

The source data conditioner is a regular n-bit data path design which consists three n-bit registers, four *n-bit* multiplexers, four n-bit adders, and one n-bit comparator. Note that n is 12 for the LJNC compressor chip design [14].

4.3 Block Adaptive Entropy Coder

The standard source pixel is then encoded by using the block adaptive entropy coding scheme. The BAEC processes input data on a block-by-block basis. A block can be any length of pixels. The fixed block size J = 16 is practical for the majority of applications.

To promote efficiency, the design of the BAEC would be made specific to the code algorithms. A functional design of the BAEC is shown in Fig. 8. The BAEC includes a Mock buffer, a code estimator, a k-bit sample splitter, and a fundamental-sequence coder. The BAEC receives the string of unsigned integers from the preprocessor and

performs coding to reduce its data rate on a block-by-block basis. Each block is made up of 16 unsigned integers. For each input block, an optimum option is chosen out of 11 optional code operators, which include the backup operator, the fundamental sequence (FS) operator, and a combination of fundamental sequence with k-bit pixel splitting, * with k = 1, 2, 3, 4, 5, 6, 7, or 8. With the chosen option, the BAEC proceeds to perform coding on the 16-sample 8-bit-per-sample input block. 1'S114 provides excellent performance over a broad range of data contents with an entropy dynamic range from 0.75 to 10.5. The eleven code options are fully implemented with only five regular hardware blocks. A brief description of each building block is as follows:

Block Buffer. The Block Buffer temporarily stores the current 16- pixel block to enable block-pipelined operation of the BAEC. At the end of an input line, if less than 16 values remain, zeros will be appended to fill the block. The resultant block will then be processed like other blocks. It is built with 16 12-bit shift registers.

Code Estimator. The code estimator chooses the code algorithm that performs best on the current block of input data. The selection is made on the bias of simple arithmetical operations on the sequence of input bits. The *Fo summer* is used to sum up all the unsigned integers in $\tilde{\delta}^n$. The set of decision regions as described in l'able 1 is realized with a random logic decoder.

k-bit Sample Splitter

The k-bit pixel splitting is implemented by using a k-bit right-logical barrel shifters.

Fundamental-Sequence Coder

The Fundamental sequence (FS) operator forms the backbone for all reversible coding operators used in the UNC except for the Backup operator. It generates a codeword for each entry value of $\tilde{\delta}^n$. For each unsigned integer i of $\tilde{\delta}^n$, its FS codeword is an $(i+1)$-bit binary string consisting of i zeroes followed by a single one. The non-zero one makes the code reversible and serves as a marker for the decoding process. This operator is also called $PSI_{1,0}[.]$ or $PSI_1[.]$. The length of a Fundamental Sequence is

$$F_n = J + \sum_{j=1}^{J} \delta_j \qquad (9)$$

where J=16 is the number of pixels in a block.

The FS coder is implemented by using a resetable and loadable memory. A biased accumulator is used to calculate the length of the sequence. The intermediate sum FO_i tracks the "single one" position of each codeword stored in the memory. The final sum FO represents the bit length of each block.

Huffman 3-tuple Coder (Ψ'_0)

The fundamental sequence coding becomes quite inefficient for low entropies, since the length of code words is at least one bit per pixel. An approach to improve the performance of the fundamental sequence coding for the low-entropy source data is achieved by a Huffman 3-tuple coding method. The redundancy left in data source $\widetilde{FS} = PSI_1[\tilde{\delta}^n]$ can be reduced by using a Huffman 3-tuple code as specified in Table 2.

The 3-tuple β_i's are derived by the following operations:

(a) Fundamental Sequence:

$$\widetilde{FS} = PSI_1[\tilde{\delta}^n] = \zeta_1\zeta_2\ldots\zeta_{F_0} \qquad (10)$$

(b) Complement:

$$\overline{\widetilde{FS}} = \bar{\zeta}_1\bar{\zeta}_2\ldots\bar{\zeta}_{F_0}$$

(c) Third Extension:

$$Ext^3[\overline{\widetilde{FS}}] = (\bar{\zeta}_1\bar{\zeta}_2\bar{\zeta}_3)^*\,(\bar{\zeta}_4\bar{\zeta}_5\bar{\zeta}_6)^*\ldots(\bar{\zeta}_{F_0}00)$$
$$= \beta_1{}^*\beta_2{}^*\ldots\beta_a$$

where a $\frac{[F_0]}{3}$ 3-*tuples*. . The last 3-tuple is filled with O if necessary.

'1'able 1. PSI14 Decision Regions, N =11, n = 8, J=16

Code Option	Code ID	Decision Region (N =11, n = 8, J =16)
Ψ'_0	0000	F0 ≤24
$\Psi'_{1,0}$	0001	24 < F0 ≤ 56
$\Psi'_{1,1}$	0010	.s6 < F0 ≤120
$\Psi'_{1,2}$	0011	120 < F0 ≤248
$\Psi'_{1,3}$	0100	***248 <HI*** ≤ .504
$\Psi'_{1,4}$	0101	504 < F0 ≤ 1016
$\Psi'_{1,5}$	0110	1016 ***<HI ≤ 2040***
$\Psi'_{1,6}$	0111	2040< F0 ≤ 408s
$\Psi'_{1,7}$	1000	***4088***< F0 ≤8184
$\Psi'_{1,8}$	1001	***8184***< F0 ≤20472
Ψ'_3	1111	***2047?*** ≤F0

Table 2 3-Tuple Huff ffman Code, cfs[i]

3-tuple β_i	code word cfs [β_i]
000	1
001	001
010	010
100	011
011	00000
101	00001
110	00010
111	00011

4.4 Compressed Data Formatter

Figure 10 depicts the format for the output bit string from the UNC coder. It consists mainly of a reference pixel and a collection of compressed data blocks. The first n-bits of the compressed output bit string is the first n-bit pixel value of the input data line and can be used as the reference pixel. The compressed bit string for each data block consists of a 4-bit *ID code* and followed by either a backup block of Jn -bit mapped differences or a *FS compressed bit string* appended with Jk bit sub-strings, where k is clctm-mined by the selected code operator $FSI_{1,k}$. The final output from the UNC after processing an input line is a compressed bit string of data which is completely packed and there are no inter-block gaps between compressed bit strings for the blocks. The compressed bit string are read out by groups of 16 bits in parallel.

The Compressed Data Formatter includes FS Packer, SS Packer, Backup Packer, and Formatter.

FS Packer, The FS Packer prepares 16-hit words for Formatter to pickup from FS sequence with ID bits.

SS Packer. The SS Packer buffers and packs k lowest-order bits of the 16 split pixels into k 16-bit words for the Formatter.

Backup Packer. If the backup code operator option is selected, Backup Packer module buffers and packs the 16 mapped differences into 16-bit words for the Formatter.

Formatter. The Formatter completely conca tenates and reads out the packed words from FS Packer, SS Packer, or Backup Packer modules to ensure that there arc no inter-block gaps between compressed bit strings. The Formatter is designed with barrel shifters and control logic gates.

4.5 Prototype Chip

The UNC compressor chip was designed, tested, and demonstrated [14]. It was designed with a 1.6-μm standard-ccl] CMOS technology and mounted in an 84-pin pin-grid-array package. As shown in Fig. 11, the UNC chip design occupies a compact chip area of 8.8 x 7.2 mm^2, with 49,000 transistors, 79 pads. The worst case power dissipation is 0,25 watts at a 10 MI Iz system clock. This compressor chip is mounted in an 84-pin pin-grid-array package. It can operate up to 20M pixels/scc. Summary of chip information is given in Table 3.

Table 3. UNC compressor chip information

Chip Name	UNC Compressor Chip
Design Method	Standard-Cell VLSI design
Process Technology	1.6-micron CMOS
Die Size	8.774 x 7.164 mm^2
Total# of Device	48,986
Total# of Cells	3??4
No. of Pads	79
Package	84-pin PGA
Power	0.25 W
Data Rate	20 M pixels/see

5 Conclusion

The goal of data compression is to red uce the communication and Storage costs for data Systems where reduction in the volume of transmitted or recorded data is important. VI .S1 data compression rc.search has been motivated by the needs of high-speed high-performance data compression and inspired by the neural networks, the parallel/pipelined processing, and the VLSI technologies. An integrated data compression system has been proposed to meet the needs of hi#~-speed high-performance data compression. Its primary application is targeted for the advanced multi-instrument spacecraft payload system that has various source data such as science data, image, video, telemetry and binary files. The proposed integrated data compression system combines a high-ratio lossy data compressor with a lossless data compressor to provide various Compression schemes to meet a broad area of applications. The embedded lossy compressor is a high-speed VLSI neuroprocessor for adaptive image compression based upon a frequency-sensitive self-organization algorithm. Performances of the

FSO neuroprocessor can achieve near-optimal results and a time complexity 0(1) for each quantization vector. A ?S-dimensional 64-codevector adaptive vector quantizer prototype chip was demonstrated to operate at 2 M vectors per second, It provides an intrinsic compression ratio as high as 33. The embedded lossless compressor is a hi~l~-speed VI .S1 pipelined processor design based on the Rice algori thin. The chip occupies a compact chip area of 8.8x7.2 mm^2 in a 1.6-µm CMOS technology and operates up to 20M pixels / see.

Acknowledgments

The research described in this paper was carried out by the Jet l'repulsion laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The author would like to thank Prof. Bing J. Sheu, Mr. Robert Rice for valuable discussions,

References

[1] D. A. Huffman, " A method for the construction of minimum redundancy codes," Poc., IRE vol. 40, pp. 1098-1101,1952.

[?1 A. K. Jain, "Image data compression: a review," Proc. of IEEE, vol. 69, no. 3, pp. 349-389, Mar. 1981.

[3] T. J. Lynch, *Data Compression- Techniques and Applications*, Lifetime Learning, Wadsworth, 1985.

[4] A . N. Netravali, B. G. Haskell, *Digital Pictures: Representat ion and Compression*, Plenum Press, 1988.

[5] J. A. Storer, *Data Compression- Methods and Theory*, Computer Science Press, Rockville, Maryland, 1988.

[6] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers: Boston, MA, 1991.

[7] B. J. Sheu, W.-C. Fang, *Image and Video Compression with VLSI Processors*, Kluwer Academic Publishers: Norwell, MA, 1993. (accepted, to be published)

[8] W . - C . Fang, *VLSI Image Compression*, Ph.D. Dissertation, University of Southern California, Los Angeles, CA, April 1992.

[9] W.-C Fang, B. J. Sheu, C). T.-C. Chen, J. Choi "A VLSI neural processor for image data compression using self-organization networks," *IEEE Trans. on Neural Network, vol.* 3, no. 3, *pp.* 506518, May 1992.

[10] Y. Linde, A. Buzo, and R, M, Gray, "An algorithm for vector quantizer design, " IEEE Trans. on Comm., vol. CC) M-28, , pp. 84-95, Jan. 1980.

[11] D. Rumelhart, J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Mlrl' Press: Cambridge, MA, 1986.

[12] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," Cog. Sci., vol.11, pp.23-63, 1987.

[13] J. -J. Lee, W.-C. Fang, R. F. Rice, "VLSI universal noiseless compressor for EOS-class missions," Proceeding of the 1988 SPIE Conference, 1 os Angeles, CA, Feb. 1988.

[14] R. Anderson, J. Bowers, W.-C. Fang, 1). Johnson, J.-J. Lee, R. Nixon, "A very high speed noiseless data compression chip for space imaging applications', Proceedings IEEE Data Compression Conference, Snowbird, Utah, April 1991.

[15] R. F. Rice, " Some practical universal noiseless coding techniques, " JPL Publication 91-3, Jet l'repulsion Laboratory, Pasadena, California, Nov. 15, 1991.

[16] R. C. Gailager "Variations on a theme by 1 luff man," IEEE Trans. on information, Theory, Vol. 1T-24, No. 6, pp. 668-674, 1978.

[17] J. H. Witten, R. M. Neal, J. G. Cleary, "Ari thmetic coding for data compression," Communications of the ACM, vol. 30, pp. 520-540.

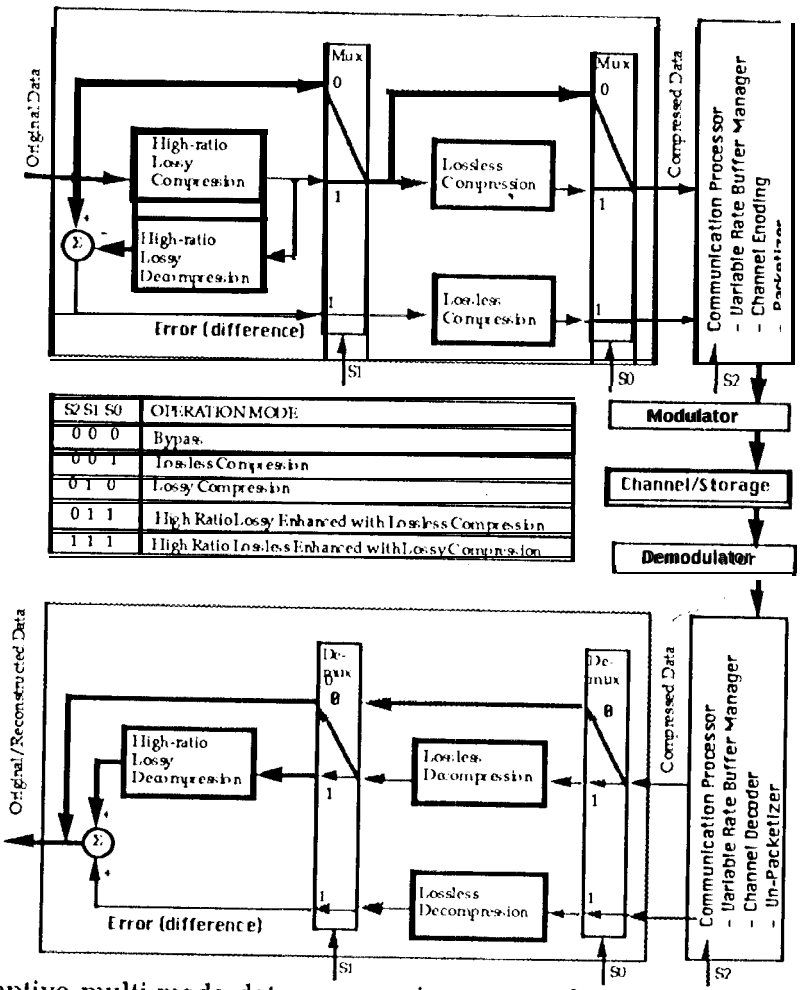
[18] T. A. Welch, " A technique for high-performance data compression," IEEE Computer Magazine, pp. 8-18,1984.

[19] J, Ziv, and A. Lempel, "Compression of individual sequences via variable-rate ceding," IEEE Trans. on information Theory, vol. 1"1'-24, 1978.

[20] 1'.-S. Yeh, R. F. Rice, W. H. Miller, "On the optimality of code options for a universal noiseless coder," JPL Publication 91-2, 1991.

[21] Request for Proposal, No. 1110-2-6562-264, for Associate Contractor for Mars Environmental Survey (MESUR) Network l'base B1 and Pathfinder Support. Technical Report, Jet Propulsion laboratory, California Institute of Technology, March 1993.

[2?] C. A. Mead, *Analog VLSI and Neural Systems*, Addison Wesley: New York, NY, 1989.



S2 S1 S0	OPERATION MODE
0 0 0	Bypass
0 0 1	Lossless Compression
0 1 0	Lossy Compression
0 1 1	High Ratio Lossy Enhanced with Lossless Compression
1 1 1	High Ratio Lossless Enhanced with Lossy Compression

Fig. 1 An adaptive multi-mode data compression system that combines a high-ratio lossy data compressor with a lossless data compressor to provide various compression schemes.

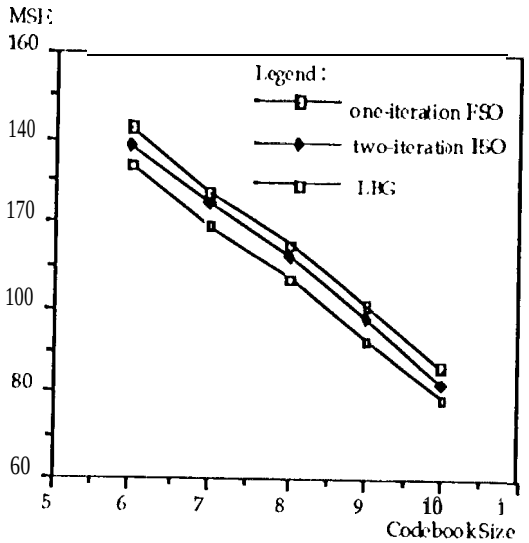


Fig. 2 Mean-squared errors of image compression using the FSO method and the LBG method on 5x5 subimage blocks of a 512x512-pixel SAR Ice image.

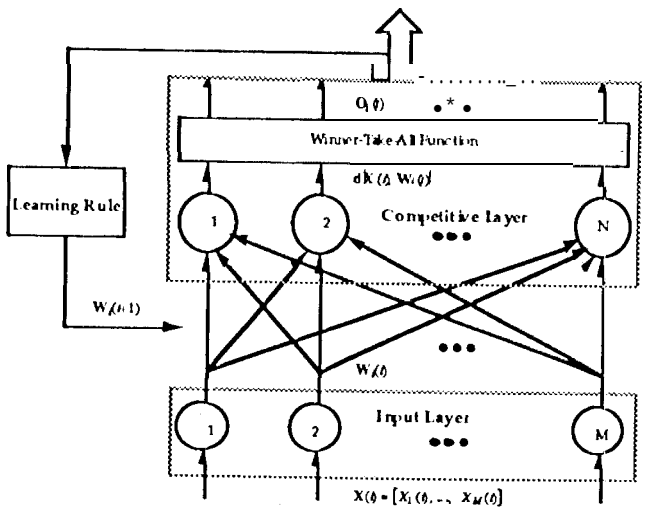


Fig. 3 Architecture of the FSO neural network.

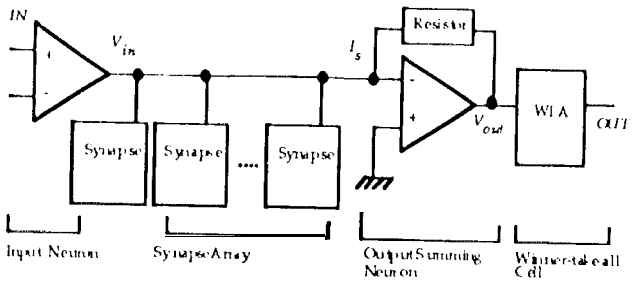


Fig. 4 Block diagram of the FSO network slice.

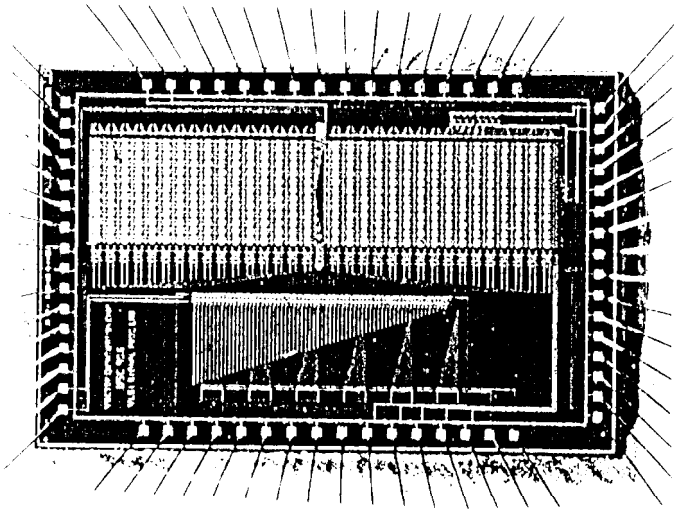


Fig. 5 A die photo of the neural-based vector quantizer prototype chip.

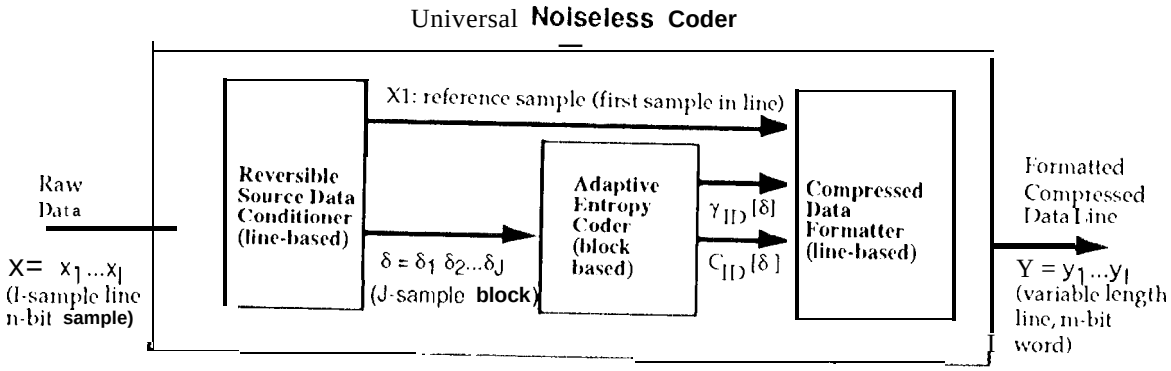


Fig. 6 A generic universal noiseless coder based on Rice algorithm.

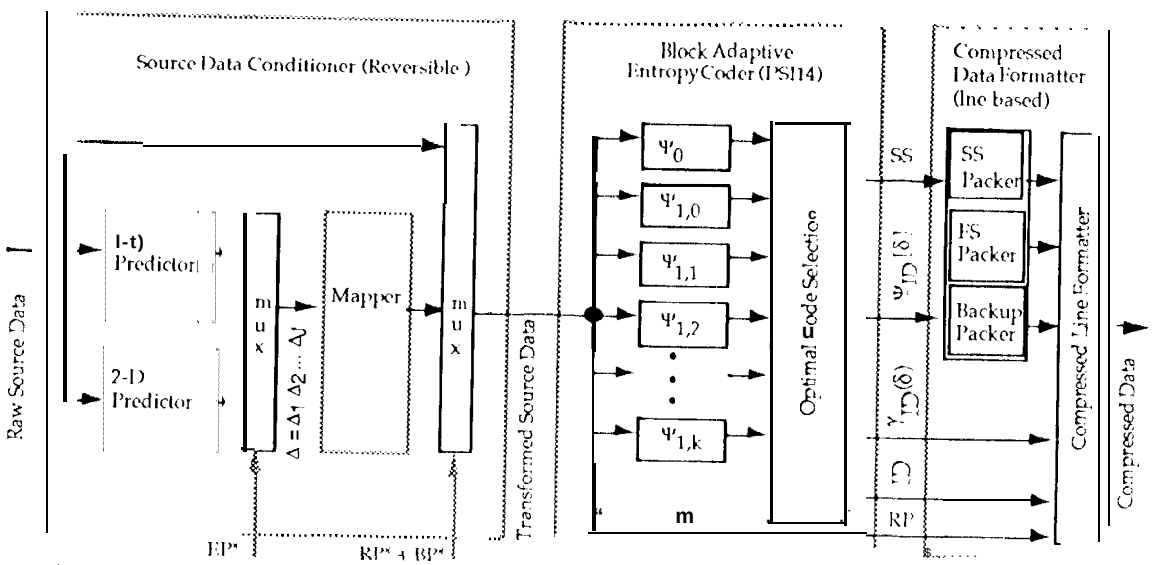


Fig. 7 Functional design of an universal noiseless coder based on Rice FSI14

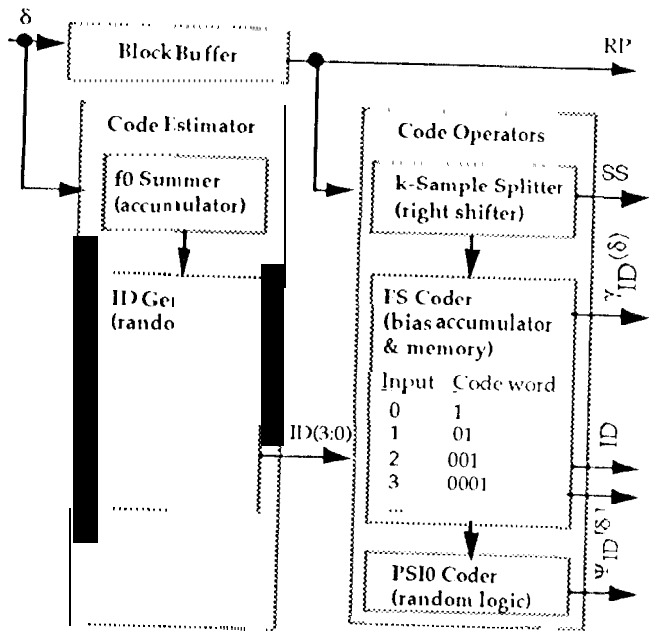


Fig. 8. Architecture design of the block adaptive entropy coder

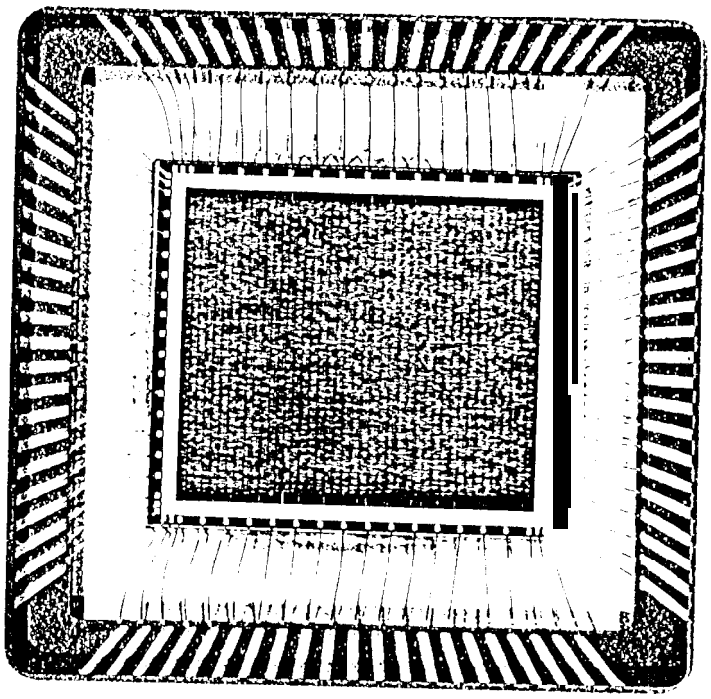


Fig. 10. Die photo of the UNC compressor chip.

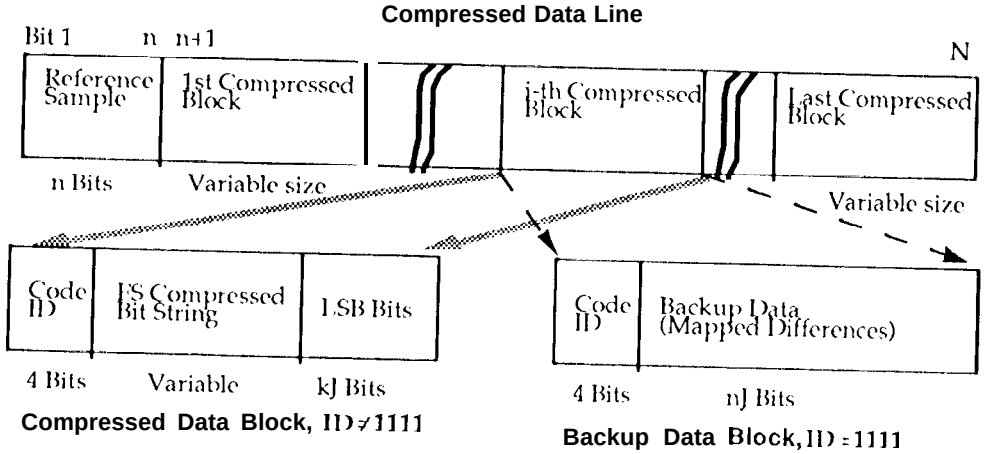


Fig. 9 Compressed line format of the UNC-PSI14.